

Meus Estudos em Análise e Desenvolvimento de Sistemas

PUC-MG

BRUNO DE M. RUAS

20 de março de 2022

Conteúdo

I	Implementação de Sistemas de Software	1
1	Projeto: Desenvolvimento Web Front-End	2
1.1	Etapa 1	2
1.2	Etapa 2	2
1.3	Etapa 3	3
1.4	Etapa 4	3
1.5	Etapa 5	3
2	Algoritmos e Abstração de Dados	5
2.1	Bibliografia	5
2.2	Estrutura de Dados Homogêneas e Heterogêneas	6
2.2.1	Estrutura de Dados Homogêneas	6
2.2.2	Estrutura de Dados Heterogêneas	6
2.3	Tipos Abstratos de Dados - Classes - Implementação	6
2.3.1	Definição de um TAD - Classes e Objetos	6
2.3.2	Atributos, Propriedades e Métodos de Classe	6
2.3.3	Mecanismos de Visibilidade/Acessibilidade	6
2.3.4	Construtores de Classe	6
3	Algoritmos e Lógica de Programação	7
3.1	Bibliografia	7
3.2	Lógica de Programação e Estrutura de Controle, Funções e Procedimentos	8
3.2.1	Conceito de Algoritmo	8
3.2.2	Variáveis	8
3.2.3	Estrutura Sequencial	8
3.2.4	Estrutura Condicional	8
3.3	Estrutura de Repetição	8
3.4	Manipulação de Dados em Memória Primária e Secundária	8
3.4.1	Criando e Usando Funções e Procedimentos	8
3.4.2	Passagem de Parâmetros	8
3.4.3	Manipulação de Arquivos em C#	8

4	Desenvolvimento Web Front-End	9
4.1	Bibliografia	9
4.2	A Web: Evolução, Padrões e Arquitetura	10
4.2.1	Histórico e Evolução da Web	10
4.2.2	W3C e os Padrões da Web	10
4.2.3	Componentes da Arquitetura da Web	11
4.2.4	URI, URL e URN	12
4.2.5	Protocolo HTTP	12
4.2.6	Servidores Web	14
4.2.7	Dinâmica de Aplicações Web	16
4.3	Desenvolvimento de Interfaces Web	17
4.3.1	A Linguagem HTML	17
4.3.2	A Linguagem CSS	19
4.3.3	A Linguagem JavaScript	19
5	Fundamentos de Engenharia de Software	20
5.1	Bibliografia	20
5.2	Conceitos e Processos de Software	21
5.2.1	Desafios e Contribuições da Área	21
5.2.2	Definições	21
5.2.3	Ciclo de Vida de Software	21
5.2.4	Processos Ágeis	21
5.2.5	Processos Prescritivos	21
5.2.6	Quando usar cada Processo?	21
5.2.7	Requisitos Funcionais	21
5.2.8	Requisitos Não Funcionais	21
5.3	Atividades e Artefatos da Engenharia de Software	21
5.3.1	Atividades Técnicas	21
5.3.2	Atividades Gerenciais	21
5.3.3	Testes de Software	21
5.3.4	Guias e Templates	21
5.3.5	Desenhando Processos de Software	21
6	Lógica Computacional	22
6.1	Bibliografia	22
6.2	Pensamento Lógico	23
6.2.1	Introdução	23
6.2.2	O que é Lógica?	23
6.2.3	Motivação	23
6.2.4	Definições	23
6.2.5	Subconjuntos	23
6.2.6	Operações sobre Conjuntos	23
6.2.7	Princípios da Lógica Proposicional	23
6.2.8	Conectivos Lógicos	23

6.2.9	Tabela Verdade e Equivalência Lógica	23
6.2.10	Predicados e Quantificadores	23
6.2.11	Ligando Variáveis	23
6.2.12	Negações	23
6.3	Pensamento Analítico	23
6.3.1	Provas de Teoremas	23
6.3.2	Regras de Inferência	23
6.3.3	Argumentos Válidos	23
6.3.4	Indução Matemática	23
6.3.5	Indução Forte	23
6.3.6	Recursão	23
6.3.7	Especificação de Sistemas	23
6.3.8	Verificação de Programas	23
7	Matemática Básica	24
7.1	Bibliografia	24
8	Organização de Computadores	26
8.1	Bibliografia	26
8.2	Fundamentos de Organização de Computadores	27
8.2.1	Representação de Dados e Sistemas Binário	27
8.2.2	Conceitos de Lógica Digital	28
8.2.3	Circuitos Lógicos Digitais Básicos	31
8.2.4	Introdução à Organização de Computadores	31
8.2.5	Unidade Central de Processamento - UCP	31
8.2.6	Memória	31
8.2.7	Entrada e Saída	31
8.3	Arquitetura de Computadores	31
8.3.1	Arquiteturas RISC e CISC	31
8.3.2	Arquitetura do Conjunto de Instruções: Exemplo do MIPS	31
8.3.3	Linguagem de Montagem	31
8.3.4	Conceito de Pipeline de Instruções	31
8.3.5	Paralelismo em Nível de Instruções	31
8.3.6	Paralelismo em Nível de Processadores	31
9	Pensamento Computacional	32
9.1	Bibliografia	32
9.2	Conceitos e Competências de Pensamento Computacional	33
9.2.1	O que é?	33
9.2.2	A Importância do Pensamento Computacional	33
9.2.3	Visão Geral	33
9.2.4	Decomposição	33
9.2.5	Reconhecimento de Padrões	33

9.2.6	Abstração	33
9.2.7	Automação	33
9.2.8	Paralelização	33
9.2.9	Simulação	33
9.2.10	Avaliação de Soluções	33
9.3	Computação Desplugada	33
9.3.1	O que é?	33
9.3.2	A importância da Computação Desplugada	33
9.3.3	Compreensão de Texto	33
9.3.4	Adivinhação de Número	33
9.3.5	Roteiro Turístico	33
9.3.6	Nonograma	33
II	Análise e Projeto de Software	34
10	Projeto: Desenvolvimento de uma Aplicação Interativa	35
11	Algoritmos e Estrutura de Dados	36
12	Desenvolvimento Web Back-End	37
13	Design de Interação	38
14	Engenharia de Requisitos de Software	39
15	Fundamentos de Redes de Computadores	40
16	Manipulação de Dados com SQL	41
17	Modelagem de Dados	42
18	Programação Modular	43
III	Processo de Negócio e Desenvolvimento de Software	44
19	Projeto: Desenvolvimento de uma Aplicação Móvel em um Ambiente de Negócio	45
20	Desenvolvimento de Aplicações Móveis	46
21	Estatística Descritiva	47
22	Gerência de Configuração	48
23	Gerência de Projetos de TI	49

24 Gerência de Requisitos de Software	50
25 Qualidade de Processos de Software	51
IV Infraestrutura para Sistemas de Software	52
26 Projeto: Desenvolvimento de um Aplicação Distribuída	53
27 APIs e Web Services	54
28 Arquitetura de Software Distribuído	55
29 Banco de Dados NoSQL	56
30 Cloud Computing	57
31 Projeto de Software	58
32 Teste de Software	59
V Empreendedorismo e Inovação com Sistemas de Software	60
33 Projeto: Desenvolvimento de um Sistema Sociotecnológico Inovador	61
34 Compliance em TI	62
35 Computadores e Sociedade	63
36 Empreendedorismo e Inovação	64
37 Implantação de Soluções de TI	65
38 Segurança Aplicada ao Desenvolvimento de Software	66

Parte I

Implementação de Sistemas de Software

Capítulo 1

Projeto: Desenvolvimento Web Front-End

1.1 Etapa 1

Objetivo da Etapa: Definir o problema a ser solucionado e os componentes do seu grupo de trabalho. Nesta etapa você entregará como tarefa dois artefatos: a **documentação de contexto** e a **especificação do projeto**.

O template da documentação do projeto pode ser baixado nesse [LINK](#)

Microfundamentos a serem estudados:

- Matemática Básica
- Pensamento Computacional
- Fundamentos de Engenharia de Software

1.2 Etapa 2

Objetivo da Etapa: Projetar a interface da aplicação e a arquitetura da solução, além de definir o ambiente de trabalho que será utilizado pela equipe para desenvolver o projeto. Os artefatos a serem produzidos são: **Projeto de Interface**, **Metodologia** e **Arquitetura da Solução**.

Microfundamentos a serem estudados:

- Fundamentos de Engenharia de Software
- Desenvolvimento Web Front-End
- Lógica Computacional

1.3 Etapa 3

Objetivo da Etapa: Desenvolver a homepage e, pelo menos, uma funcionalidade da solução projetada. O primeiro artefato a ser gerado é o **Template do site**, que determina o layout padrão do site (HTML e CSS) que será utilizado em todas as páginas com a definição de identidade visual, aspectos de responsividade e iconografia. No desenvolvimento das funcionalidades, cada artefato gerado (código fonte) deve estar relacionado a um requisito funcional e/ou não funcional.

Microfundamentos a serem estudados:

- Desenvolvimento Web Front-End
- Algoritmos e Lógica de Programação

1.4 Etapa 4

Objetivo da Etapa: Finalizar o desenvolvimento da solução e irá elaborar e executar o plano de testes funcionais. Os artefatos serão: O **plano de testes de software** e O **Registro de Testes de Software**.

Microfundamentos a serem estudados:

- Algoritmos e Lógica de Programação
- Fundamentos de Engenharia de Software
- Algoritmos e Abstração de Dados

1.5 Etapa 5

Objetivo da Etapa: Apresentar a versão final da solução implantada.

A apresentação do projeto consiste na geração de um conjunto de slides em um arquivo no formato ppt, pptx ou pdf, contemplando os seguintes itens:

- Contexto (Problema, Público-alvo);
- Requisitos;
- Solução Implementada (funcionalidades de software);
- Conclusão da elaboração do projeto (pontos positivos, desafios, aprendizado).

CAPÍTULO 1. PROJETO: DESENVOLVIMENTO WEB FRONT-END4

Recomenda-se não ultrapassar 10 slides, pois o tempo de apresentação é limitado a 10 minutos, sendo 5 minutos para o projeto (slides) e 5 minutos para a demonstração da aplicação.

A equipe também deverá gravar um vídeo de, no máximo, três minutos, com a apresentação da solução. Vocês deverão abrir a aplicação hospedada e apresentar o seu funcionamento.

Microfundamentos a serem estudados:

- Fundamentos de Engenharia de Software
- Organização de Computadores

Capítulo 2

Algoritmos e Abstração de Dados

2.1 Bibliografia

Bibliografia Básica

- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. Fundamentos da programação de computadores. São Paulo: Pearson, 2012. ISBN 9788564574168
- SOUZA, Marco A. Furlan de; GOMES, Marcelo Marques; SOARES, Marcio Vieira; CONCÍLIO, Ricardo. Algoritmos e lógica de programação: um texto introdutório para a engenharia. São Paulo: Cengage Learning, 2019. ISBN: 9788522128150
- ACM TRANSACTIONS ON PROGRAMMING LANGUAGES AND SYSTEMS. New York: Association for Computing Machinery.,1979-. 6 times a year. Absorvido ACM letters on programming languages and systems. ISSN 0164-0925. Disponível em: <https://dl-acm-org.ez93.periodicos.capes.gov.br/citation.cfm?id=J783>
- AGUILAR, Luis Joyanes. Fundamentos de programação algoritmos, estruturas de dados e objetos. 3. ed. Porto Alegre: AMGH, 2008. ISBN: 9788580550146

Bibliografia Complementar

- DEITEL, Harvey M; DEITEL, Paul J. Java - como programar. 8. ed. São Paulo: Pearson, 2010. ISBN 9788576055631
- GRIFFITHS, Ian. Programming C# 8.0. O'Reilly Media, Inc. 2019. ISBN 9781492056812

- MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. Algoritmos: lógica para desenvolvimento de programação de computadores. 28. ed. rev. e atual. São Paulo, SP: Érica, 2016. E-book. ISBN 9788536518657
- PROGRAMMING AND COMPUTER SOFTWARE. New York, Consultants Bureau, 1975-. Bimestral. ISSN 1608-3261. Disponível em: <https://link-springer-com.ez93.periodicos.capes.gov.br/journal/volumesAndIssues/11086>
- PRICE, Mark J. C# 8.0 and .NET Core 3.0 - Modern Cross - Platform Development. O'Reilly Media; 2019. ISBN 9781788478120
- PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estruturas de dados com aplicações em Java. 2. ed. São Paulo: Prentice Hall, 2009. ISBN 9788576052074

2.2 Estrutura de Dados Homogêneas e Heterogêneas

2.2.1 Estrutura de Dados Homogêneas

2.2.2 Estrutura de Dados Heterogêneas

2.3 Tipos Abstratos de Dados - Classes - Implementação

2.3.1 Definição de um TAD - Classes e Objetos

2.3.2 Atributos, Propriedades e Métodos de Classe

2.3.3 Mecanismos de Visibilidade/Acessibilidade

2.3.4 Construtores de Classe

Capítulo 3

Algoritmos e Lógica de Programação

3.1 Bibliografia

Bibliografia Básica

- Ana Fernanda Gomes ASCENCIO; Edilene Aparecida Veneruchi de CAMPOS. Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java - 2ª edição. São Paulo, SP : Pearson Education do Brasil, 2012

Bibliografia Complementar

- H. DEITEL et. Al. C#: Como Programar. São Paulo: Makron Books, 2003
- John SHARP. Microsoft Visual C# 2013. Grupo A, 2014
- André Luiz Villar FORBELLONE, Henri Frederico EBERSPÄCHER. Lógica de programação: a construção de algoritmos e estruturas de dados. São Paulo: Prentice Hall, 2005.
- MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. Algoritmos: lógica para desenvolvimento de programação de computadores 28. ed. rev. e atual. São Paulo, SP: Érica, 2016
- Sandra PUGA, Gerson RISSETTI. Lógica de Programação e Estrutura de Dados: com aplicações em Java - 2ª edição. São Paulo : Pearson, 2017

3.2 Lógica de Programação e Estrutura de Controle, Funções e Procedimentos

3.2.1 Conceito de Algoritmo

3.2.2 Variáveis

3.2.3 Estrutura Sequencial

Etapas de um Algoritmo e o Operador de Atribuição

Operadores e Funções Aritméticas

Expressões Aritméticas

3.2.4 Estrutura Condicional

Condição Simples e Composta

Operadores Booleanos e Exemplos de Uso do Comando IF

O Comando Switch e o Operador Ternário

3.3 Estrutura de Repetição

Os Comandos WHILE, DO WHILE e FOR

Contadores e Acumuladores

3.4 Manipulação de Dados em Memória Primária e Secundária

3.4.1 Criando e Usando Funções e Procedimentos

3.4.2 Passagem de Parâmetros

3.4.3 Manipulação de Arquivos em C#

Capítulo 4

Desenvolvimento Web Front-End

4.1 Bibliografia

Bibliografia Básica

- SIKOS, L. Web Standards. Mastering HTML5, CSS3, and XML.
- DACONTA, M. C.; SMITH, K. T.; OBRST, L. J. The semantic Web: a guide to the future of XML, Web services, and knowledge management. [s. l.]: Wiley, [s. d.]. ISBN 0471432571
- SILVA, Maurício Samy. HTML 5: a linguagem de marcação quer revolucionar a web. 2. ed. rev. e ampl. [s. l.]: Novatec, 2014. ISBN 9788575224038
- SANDERS, William B. Smashing HTML5: técnicas para a nova geração da web. Porto Alegre: Bookman, 2012. xiv, 354 p. ISBN 9788577809608
- DEITEL, Paul J., Deitel, Harvey M. Ajax, Rich. Internet Applications e Desenvolvimento Web para Programadores. Pearson 776. ISBN 9788576051619
- SILVA, Maurício Samy. CSS3: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3. São Paulo: Novatec, 2011. 494 p. ISBN 9788575222898
- BERTAGNOLLI, S. de C.; MILETTO, E. M. Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, JavaScript e PHP. [s. l.]: Bookman, 2014. ISBN 9788582601952

4.2 A Web: Evolução, Padrões e Arquitetura

4.2.1 Histórico e Evolução da Web

A Web é um sistema da informação de hipertextos onde o acesso é feito por meio de **navegadores (browsers)**.

Existem alguns protocolos comuns para transferência de alguns tipos de arquivos. Para mensagens (e-mail) usamos o **SMTP**, para transferência de arquivos usamos o **FTP**, aplicações de telefonia usam o **VOIP** e para páginas de conteúdo usamos o **HTML**.

A história da web eu ainda vou colocar aqui quando tiver mais tempo.

4.2.2 W3C e os Padrões da Web

O WORLD WIDE WEB CONSORTIUM (W3C) é uma organização sem fins lucrativos cujo líder é o Tim Berners-Lee, justamente o inventor da Web. Existem várias organizações ao longo do planeta que fazem parte desse consórcio internacional.

O W3C mantém a gestão de vários padrões usados todos os dias:

- Design e Aplicações Web (HTML, CSS, SVG, Ajax, Acessibilidade);
- Arquitetura da Web (Protocolo HTTP, URI);
- Web Semântica (Linked Data - RDF, OWL, SPARQL);
- Web Services (SOAP, WSDL);
- Tecnologia XML (XML, XML Schema, XSLT);
- Navegadores e ferramentas de autoria.

A W3C possui um processo de publicação das normativas. Normalmente, o fluxo é:

- Working Draft (WD)
- Candidate Release (CR)
- Proposed Recommendation (PR)
- Recommendation (REC)

4.2.3 Componentes da Arquitetura da Web

a web pode ser entendida como uma coleção de componentes que permitem a comunicação entre o cliente e os servidores de aplicações. Os principais componentes dessa arquitetura são:

- Ambiente Cliente (Client Web)

Geralmente um Browser que envia as requisições usando o protocolo HTTP(S) para o servidor web através de uma rede de computadores.

- Ambiente Servidor

O ambiente servidor possui vários componentes (banco de dados, aplicações, API e etc) mas o principal componente é o servidor web. Ele recebe a requisição HTTP(S) do client, interpreta a URL e envia os recursos solicitados (HTML, CSS, JS, JPEG, MP4 e etc) por meio da rede.

- Internet

É a rede mundial baseada no protocolo TCP/IP onde todo computador conectado é denominado host (hospedeiro) e possui um identificador de endereço IP (internet protocol) que possui determinados padrões.

- URI (uniform resource locator)

Como o nome indica, um URI é um localizado que pode ser classificado em duas maneiras. O URL é o tipo de URI que usa o endereço do conteúdo como método de localização, ele nos diz onde encontrar o recurso (por exemplo, o caminho `c://home/desktop/test.txt`). O URN é o tipo que usa o nome do recurso, ele nos diz a identidade do item procurado (por exemplo, o sistema ISBN).

- Requisição

É o pacote de dados enviado pelo client através da internet para o web server onde está a instrução do que deve ser enviado como resposta.

- Resposta

Como o nome já diz, é o retorno do web server ao client com os dados requisitados.

- Protocolo HTTP

É o padrão como client e web server se comunicam pela rede.

4.2.4 URI, URL e URN

Já vimos que o URI abarca dos conceitos de URL e URN. Agora vamos aprender um pouco mais sobre os padrões de endereços em ambos os protocolos.

URL

O padrão URL serve para identificar o recurso pela sua localização e é composto da seguinte maneira:

```
ftp://example.com:8080/pasta/arquivo?name=book#nose
      Cujas partes são
scheme://authority/path?query#fragment
```

Como podemos ver, a URL é composta por várias partes:

- scheme - é a forma de interação (ftp, http, https, ...).
- user:pass - são as informações do user.
- host - endereço de ip do server.
- porta - qual a porta TCP/IP do server (o padrão http é 80 e pode ser omitida).
- path - local onde o recurso se encontra.
- query - detalhe da consulta na forma de pares nome-valor.
- fragmento - qual seção do recurso.

URN

```
urn:example:animal:ferret:nose
      Cujas partes são
scheme:path:authority
```

A URN apenas nos dá um recurso específico (NSS) contido em algum namespace (NID) sem qualquer informação sobre onde o arquivo está localizado.

4.2.5 Protocolo HTTP

O hypertext transfer protocol é mantido pela W3C e rege a camada de aplicação dos sistemas distribuídos de informação em hipertexto. Existem muitas versões mas a mais utilizada é a 2.0 de 2015.

Para entender melhor o http, consideremos o processo usual de navegação na web:

1. user informa a URL
2. client monta a requisição http e envia ao web server
3. server recebe a requisição e envia a resposta ao client
4. a resposta é recebida e interpretada pelo browser com os dados exibidos ao user
5. dependendo da página, pode ser que novas requisições sejam feitas para que todos os componentes sejam carregados propriamente.

Podemos ver que o http é o conjunto de regras que rege a comunicação client-server da web.

Partes da requisição HTTP

Uma requisição é formada por 3 partes:

- Linha de Requisição
 - Método
 - * GET - Requisita dados.
 - * POST - Envia dados para o server.
 - * HEAD - Requisita dados mas o retorno deve ser um conjunto de cabeçalhos.
 - * PUT - Criação ou Atualização de dados.
 - * DELETE - Excluir algum dado.
 - * TRACE - Solicita uma cópia da requisição (serve pra testar integridade).
 - * PATCH - Alterações parciais em um recurso.
 - * OPTIONS - Lista de métodos e opções disponíveis para o server.
 - * CONNECT - Usado quando o client se conecta com o server via proxy.
 - Recurso - É o caminho do dado requerido.
 - Versão do Protocolo - Qual versão do http será usada.
- Linhas de Cabeçalho
 - Inclui informações complementares à requisição sendo formado por pares nome-valor.
- Corpo da Entidade
 - Dados adicionais como forms, arquivos para upload e etc.

Partes da resposta HTTP

- Linha de Resposta
 - Versão HTTP.
 - Código de Retorno.
 - Mensagem explicativa do código de retorno.
- Linhas de Cabeçalho

Uma informação importante que consta no cabeçalho é o `content-type`. Ele diz o formato do conteúdo enviado como resposta¹. Essa informação é apresentada conforme os MIME Types.
- Corpo da Entidade É o recurso solicitado pelo client (html, css, js, jpeg, mp4, ...)

Um pouco mais sobre Cabeçalhos

Os cabeçalhos possuem características parecidas tanto nas requests quanto nas responses. Podemos classifica-los como sendo dos tipos:

- Request header - Informações sobre o client ou a requisição feita.
- Response header - Informações sobre a resposta ou sobre o web server.
- Entity header - Informações sobre o conteúdo da entidade trocada (tamanho e tipo).
- General header - Informações gerais usadas tanto em requests quanto em responses.

4.2.6 Servidores Web

Você só consegue visualizar as informações de um site porque o servidor web foi capaz de interpretar a requisição feita pelo seu browser e responder com o conteúdo adequado. Agora vamos entender um pouco mais sobre o ambiente servidor.

O principal elemento do ambiente servidor é o web server. Ele é quem recebe, interpreta e responde as requisições dos clients ao longo da internet. Podemos também incluir outros elementos importantes no ambiente servidor como o **servidor de banco de dados** e os **servidores de serviços (APIs)**.

¹ Isso pode ser a causa de alguns bugs na sua aplicação.



Figura 4.1: Esboço da arquitetura web

Funções de um web service

Um web service nada mais é que um software rodando em uma máquina. Ele desempenha várias funções que podemos elencar como:

- Atender as requests http e responder a elas.
- Gerenciar sites.
- Gerenciar arquivos dos sites.
- Integrar mecanismos de scripts: php, perl, aspx, Ruby, Python e etc.
- Autenticar users (básica ou com servidores de autenticação).
- Implementar criptografia nas comunicações (https - tls/ssl).
- Cache de recursos.
- Auditoria das alterações e logs.

Software e Provedores

Basicamente, existem 3 formas de tornar uma aplicação web acessível aos clients: Rodar um web server na máquina local; instalar e configurar um web server em uma máquina dedicada para esse trabalho e, por fim, contratar um provedor que ofereça esse serviço.

A lista de softwares que se propõe a fazer o trabalho de um web server é enorme. O material do curso elenca dois:

- Apache HTTP Server | Apache Web Server
É um open source multi plataforma. Permite execução de multilinguagens como php, perl entre outras. Uma maneira simples de instalar

é pelo XAMPP (que já integra o apache web server, banco de dados MariaDB e um ambiente PHP e Perl).

- Microsoft Internet Information Server (IIS)
É a solução proprietária da Microsoft. Baseado na plataforma .NET, permite hospedar sites estáticos. O IIS já vem disponível junto dos SO Windows.

A lista de provedores também é extensa e possuem diferentes capacidades distintas mas podemos destacar algumas ferramentas úteis:

- Servidores em Nuvem
 - Azure
 - Heroku²
 - AWS
- Editores e IDEs online
 - Replit
 - CodeSandbox
 - Glitch
 - GitHub Pages

4.2.7 Dinâmica de Aplicações Web

Quando você acessa um site, o arquivo que coordena o modo de exposição da informação e os conteúdos da mesma é um arquivo “.html”. Observe o exemplo abaixo de uma página simples.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <script> src='app.js'</script>
  </head>
  <body>
    <img src='logo.jpg' alt="imagem_logo">
  </body>
</html>
```

²Esse aqui eu to estudando e fazendo um manual de como usar. Você pode ler o manual nesse [LINK]

As tags que contém as partes `style.css`, `app.js` e `logo.jpg` fazem menção à outros arquivos que farão parte da composição da página. Alguns são referentes à funcionalidades ou layout da aplicação enquanto outros podem ser referentes à conteúdos mostrados na página.

Uma vez que o servidor compreende a request feita pelo client, ele envia uma série de arquivos que serão lidos pelo browser do usuário e serão interpretados por ele. O html é justamente o primeiro arquivo lido porque ele diz ao navegador quais conteúdos mostrar e, a partir das referências contidas no html, como mostrar e quais funcionalidades a página terá.

O processamento de um site

1. O client envia uma requisição via http (com o método GET) para o web server
2. O server envia o arquivo html da página requisitada para o browser
3. Ao processar o html, o browser percebe que ele faz menção de outros arquivos (como css, js, mp3, etc)
4. O browser faz novas requisições ao server até ter todos os arquivos necessários para o carregamento da página

Como você pode ver, é muita coisa acontecendo. Você só não se dá conta disso porque o processo é muito rápido hoje em dia devida a velocidade das nossas conexões banda larga. Lembrando sempre que todas as requisições e respostas entre client e server são feitas usando-se o protocolo HTTP que a gente viu logo antes.

4.3 Desenvolvimento de Interfaces Web

4.3.1 A Linguagem HTML

A linguagem HTML foi criada por Tim Berners-Lee no ano de 1991 e foi baseada no padrão Standard Generalized Markup Language (SGML). Seu escopo original era para permitir a divulgação de pesquisas científicas.

Com o passar dos anos, novas tecnologias foram somadas ao ecossistema para facilitar o processo de construção das soluções web. O Cascading Style Sheet (CSS) foi criado para facilitar o desenvolvimento do conteúdo separando a parte de estilo e aparência do conteúdo em HTML. O JavaScript permitiu a manipulação de elementos além de dar mais dinâmica para as páginas web.

O W3C foi criado em 1993 e, a partir dessa data, o HTML foi mantido e padronizado por essa organização. Desde então a linguagem vem sendo alterada para permitir sua evolução.

Em 2004 foi criado o Web Hypertext Application Technology Working Group (WHATWG) por pessoas da Apple, Mozilla e Opera. Na época, o W3C estava trabalhando no padrão XHTML 2.0 (que iria substituir o HTML 4.01) mas o WHATWG conseguiu propor um monstro que acabou sendo o HTML 5. O HTML 5 foi recebido e amplamente adotado no desenvolvimento de aplicações hoje em dia.

Panorama de uma aplicação

Nós já sabemos que um client faz uma requisição ao web server por HTTP e esse, por sua vez, responde a requisição com, normalmente, um arquivo HTML. De posse de arquivo, o browser consegue saber se precisará solicitar mais arquivos ao web server até que todas as referências do HTML sejam satisfeitas e a página carregada.

A grosso modo, podemos dizer que o HTML pode fazer menções a arquivos dos seguintes tipos:

- CSS
- Arquivos de Multimídia
- JavaScript
- RIA - Rich Internet Applications
 - Applet Java
 - Adobe Flash
 - Adobe Air
 - Adobe Flex
 - SilverLight

Se o site utiliza soluções dinâmicas como PHP, Java, Python, Ruby ou ASP.NET, quando a requisição é feita, o web server primeiro faz o processamento desses arquivos (normalmente por um outro servidor de APIs) e o resultado serão outros arquivos HTML, CSS, JS ou Multimídia. Após o processamento, o resultado é enviado para o client que será atualizado pelo browser.

Elementos de Texto e Multimídia

Elementos Estruturais

Elementos de Tabelas

Elementos de Formulários

4.3.2 A Linguagem CSS

Seletores de Elementos

Valores e Unidades

Display e Box Model

Propriedades de Texto

Layouts Responsivos

Frameworks front-end - Bootstrap

4.3.3 A Linguagem JavaScript

Variáveis e Tipos de Dados

Controle de Fluxo

Funções

Documento Object Model (DOM)

A Notação de Objetos (JSON)

Programação Ajax

Capítulo 5

Fundamentos de Engenharia de Software

5.1 Bibliografia

Bibliografia Básica

- PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016. E-book ISBN 9788580555349. Capítulos 1, 2, 3
- PRIKLADNICKI, Rafael, WILLI, Renato, e MILANI, Fabiano. Meé-todos ágeis para desenvolvimento de software. Porto Alegre: Bookman, 2014 1 recurso online ISBN 9788582602089 Capítulos 1,2,3,8,12,13
- SOMMERVILLE, Ian. Engenharia de software, 10ª ed. Pearson 768 ISBN 9788543024974 Capítulos 1,2,3,4

Bibliografia Complementar

- COHN, Mike; SILVA, Aldir José Coelho Corrêa da. Desenvolvimento de software com Scrum: aplicando métodos ágeis com sucesso. Porto Alegre: Bookman, 2011. E-book ISBN 9788577808199
- LARMAN, Craig. Utilizando UML e padrões: uma introdução á análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman, 2007. E-book (695 páginas) ISBN 9788577800476
- PAULA FILHO, Wilson de Pádua. Engenharia de software, v. 2 projetos e processos. 4. Rio de Janeiro LTC 2019 1 recurso online ISBN 9788521636748

- VETORAZZO, Adriana de Souza. Engenharia de software. Porto Alegre SAGAH 2018 1 recurso online ISBN 9788595026780
- WAZLAWICK, Raul Sidnei. Engenharia de software conceitos e práticas. Rio de Janeiro GEN LTC 2013 1 recurso online ISBN 9788595156173

5.2 Conceitos e Processos de Software

5.2.1 Desafios e Contribuições da Área

5.2.2 Definições

5.2.3 Ciclo de Vida de Software

5.2.4 Processos Ágeis

5.2.5 Processos Prescritivos

5.2.6 Quando usar cada Processo?

5.2.7 Requisitos Funcionais

5.2.8 Requisitos Não Funcionais

5.3 Atividades e Artefatos da Engenharia de Software

5.3.1 Atividades Técnicas

5.3.2 Atividades Gerenciais

5.3.3 Testes de Software

5.3.4 Guias e Templates

5.3.5 Desenhando Processos de Software

Capítulo 6

Lógica Computacional

6.1 Bibliografia

Bibliografia Básica

- HUNTER, David J. Fundamentos de Matemática Discreta. Rio de Janeiro: LTC, 2011

Bibliografia Complementar

- ROSEN, Keneth H. Discrete Mathematics and its Applications. New York: McGraw-Hill, 2019

6.2 Pensamento Lógico

6.2.1 Introdução

6.2.2 O que é Lógica?

6.2.3 Motivação

6.2.4 Definições

6.2.5 Subconjuntos

6.2.6 Operações sobre Conjuntos

6.2.7 Princípios da Lógica Proposicional

6.2.8 Conectivos Lógicos

6.2.9 Tabela Verdade e Equivalência Lógica

6.2.10 Predicados e Quantificadores

6.2.11 Ligando Variáveis

6.2.12 Negações

6.3 Pensamento Analítico

6.3.1 Provas de Teoremas

6.3.2 Regras de Inferência

6.3.3 Argumentos Válidos

6.3.4 Indução Matemática

6.3.5 Indução Forte

6.3.6 Recursão

6.3.7 Especificação de Sistemas

6.3.8 Verificação de Programas

Capítulo 7

Matemática Básica

Como o escopo dessa matéria é super básico. Eu nem vou me dar o trabalho de resumir. Se quiserem ver um material mais completo, podem conferir na Bibliografia ou no meu Projeto Matemática.

7.1 Bibliografia

Bibliografia Básica

- GERSTING, Judith L. Fundamentos matemáticos para a ciência da computação. 7. Rio de Janeiro LTC 2016 1 recurso online ISBN 9788521633303
- HUNTER, David J. Fundamentos de matemática discreta. Rio de Janeiro LTC 2011 1 recurso online ISBN 9788521635246
- LIMA, Diana Maia de. Matemática aplicada à informática. Porto Alegre Bookman 2015 1 recurso online ISBN 9788582603178
- STEWART, James. Cálculo, v. 1. 8.ed. São Paulo (SP): Cengage Learning, 2017 E-book ISBN 9788522126859

Bibliografia Complementar

- MENEZES, Paulo Blauth. Aprendendo matemática discreta com exercícios, v.19. Porto Alegre Bookman 2011 ISBN 9788577805105
- REVISTA DE INFORMÁTICA TEÓRICA E APLICADA. Porto Alegre: UFRGS, Instituto de informação, 1989. ISSN 0103-4308
- ROSEN, Kenneth H. Matemática discreta e suas aplicações. Porto Alegre ArtMed 2010 ISBN 9788563308399
- SIMÕES-PEREIRA, José Maunel dos Santos. Introdução à Matemática Combinatória. Editora Interciência 338 ISBN 9788571932920

- ÁVILA, Geraldo; ARAÚJO, Luis Cláudio Lopes de. Cálculo: ilustrado, prático e descomplicado. Rio de Janeiro, RJ: LTC - Livros Tecnicos e Científicos, 2012. E-book ISBN 978-85-216-2128-
- GUIDORIZZI, Hamilton Luiz. Um curso de cálculo, v. 1. 6. Rio de Janeiro LTC 2018 1 recurso online ISBN 9788521635574

Capítulo 8

Organização de Computadores

8.1 Bibliografia

Bibliografia Básica

- STALLINGS, William. Arquitetura e organização de computadores. 10. ed. São Paulo: Pearson, c2018. E-book. ISBN 9788543020532
- CORRÊA, Ana Grasielle Dionísio (Org.). Organização e arquitetura de computadores. São Paulo: Pearson, 2017. E-book. ISBN 9788543020327
- PATTERSON, David A. Organização e projeto de computadores a interface hardware/software. Rio de Janeiro, GEN LTC 2017. 1 recurso online. ISBN 9788595152908

Bibliografia Complementar

- TANENBAUM, Andrew S.; AUSTIN, Todd. Organização estruturada de computadores. 6. ed. São Paulo, SP: Pearson Education do Brasil, 2013. E-book. ISBN 9788581435398
- MONTEIRO, Mário A. Introdução à organização de computadores. 5. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos, c2007. E-book. ISBN 978-85-216-1973-4

8.2 Fundamentos de Organização de Computadores

8.2.1 Representação de Dados e Sistemas Binário

Compreendendo o Sistema Decimal

Os componentes eletrônicos digitais só permitem dois estados de tensão: 0 e 1. Isso implica que toda informação manipulada pelos computadores é representado em um sistema de **numeração binária**.

O nosso modelo de sistema numérico usual é o decimal (também chamado base 10). Ele é um sistema posicional porque o peso do dígito é dependente da posição dele no número. Por exemplo:

$$38_{10} = 3 \times 10^1 + 8 \times 10^0 = 30 + 8$$
$$17,25_{10} = 10 \times 10^1 + 7 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$$

O subscrito indica o tipo de base usado. Uma característica dos sistemas posicionais é que o dígito mais a esquerda será o mais significativa (MSB - Most Significant Bit) e os à sua direita serão os LSB (Less Significant Bit).

O Sistema Binário

Como você já deve saber, no sistema binário temos apenas dois símbolos, entretanto, podemos representar todos os dígitos através deles só vamos precisar de mais dígitos binários (Binary Digit - BIT). Agora vamos aprender como representar números maiores que 1 em um sistema base 2.

O sistema binário é um sistema posicional (guarde isso na sua memória). Então a lógica é a mesma que os exemplos acima em base 10 mas com a diferença de multiplicarmos os números por 2^n onde n é a posição do dígito. Nos número fracionais é igual ao sistema base 10, basta multiplicarmos as posições dos dígitos após a vírgula por números negativos da esquerda para direita.

Para converter um número base 10 em binário você precisa estar com a lista de potências de 2 na ponta da língua. Basta ir "caminhando" por ela até o maior valor abaixo do número desejado. Essa posição será o seu MSB = 1. Depois, basta ir calculando o quanto lhe falta para obter o número desejado. Abaixo nós representamos os mesmo números da seção de números base 10.

$$38_{10} = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 100110_2$$

$$17,25_{10} = 1 \times 2^5 + 0 + 0 + 0 + 1 \times 2^0, 0 \times 2^{-1} + 1 \times 2^{-2} = 1001,01_2$$

O Sistema Hexadecimal

O sistema hexadecimal por sua vez possui 16 símbolos (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) e pode ser convertido mais facilmente em binário que o sistema base 10. Assim a gente não precisa ficar trabalhando como binário (que acaba usando muitos dígitos para representar algum valor).

A conversão é feita pela equivalência entre cada 4 dígitos binários relacionados a cada símbolo hexadecimal de acordo com a tabela abaixo

Binário	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Por exemplo, vamos converter um número hexadecimal em binário. Usando a tabela acima, podemos pegar cada dígito (1,7,F) e substituir pelo seu valor binário. Após isso, podemos descartar os zeros a esquerda do MSB.

$$17F_{16} = \underset{1}{0001} \underset{7}{0111} \underset{F}{1111} = 10111111_2$$

Para converter de binário para hexadecimal é só começar da direita para a esquerda em cada grupo de 4 bits.

8.2.2 Conceitos de Lógica Digital

Computadores são formados por componentes eletrônicos. Os transistores e os diodos são usados para a construção das portas lógicas que nos permitem, através de circuitos elétricos, replicar os operadores lógicos da lógica usados

na algebra booleana. Assumindo valores em dois estágios: 0 (de 0 a 0,6 volts) e 1 (entre 3,6 e 5 volts).

Uma porta lógica nada mais é que um circuito que recebe sinais de entrada e, conforme a sua configuração, produz um sinal de saída cujo valor é dependente da entrada.

Podemos categorizar as portas lógicas em 3 grupos:

- Portas Lógicas Básicas
 - Operação Lógica - AND
 - Operação Lógica - OR
 - Operação Inversora - NOT
- Funções e Portas Lógicas Compostas
 - Operação Lógica - NAND (NOT-AND)
 - Operação Lógica - NOR (NOT-OR)
 - Operação Lógica - XOR (OR-EXCLUSIVA)
- Expressões Lógicas e Circuitos Digitais

Eu já trabalhei bem a fundo a lógica matemática no meu curso do Projeto Matemática. Você pode conferir no capítulo 02 nesse [\[LINK\]](#). A única diferença é que quando lá for TRUE ou VERDADE, aqui será 1 e, claramente, quando lá for FALSE ou FALSO, aqui será 0.

Como nosso estudo nesse manual é mais focado em ADS eu só vou manter as anotações referentes à transposição da algebra booleana para os circuitos eletrônicos.

Operadores Básicos



Figura 8.1: Porta NOT | Equação $Y = \bar{A}$

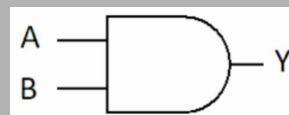
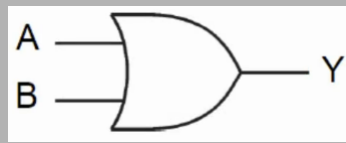
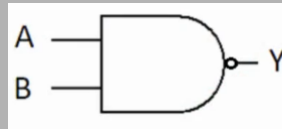
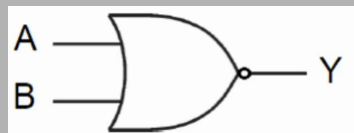
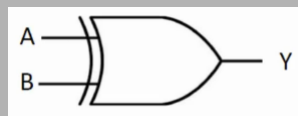
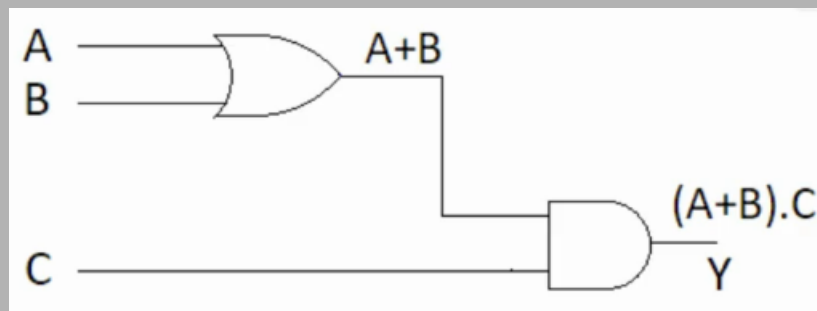


Figura 8.2: Porta AND | Equação $Y = A \cdot B$

Figura 8.3: Porta OR | Equação $Y = A + B$ **Operadores Compostos**Figura 8.4: Porta OR | Equação $Y = \overline{A + B}$ Figura 8.5: Porta OR | Equação $Y = \overline{A + B}$ Figura 8.6: Porta OR | Equação $Y = A \oplus B$ **Expressões Lógicas e Circuitos**

Podemos usar os operadores lógicos para criar expressões do tipo $Y = (A + B) \cdot C$ que pode ser lida como "Y é igual a A ou B e C"¹. Podemos também usar os diagramas de circuitos para representar exatamente essa mesma operação lógica.

¹Usando os símbolos lógicos mais clássicos, podemos escrever como $Y : (A \vee B) \wedge C$



8.2.3 Circuitos Lógicos Digitais Básicos

8.2.4 Introdução à Organização de Computadores

8.2.5 Unidade Central de Processamento - UCP

8.2.6 Memória

8.2.7 Entrada e Saída

8.3 Arquitetura de Computadores

8.3.1 Arquiteturas RISC e CISC

8.3.2 Arquitetura do Conjunto de Instruções: Exemplo do MIPS

8.3.3 Linguagem de Montagem

8.3.4 Conceito de Pipeline de Instruções

8.3.5 Paralelismo em Nível de Instruções

8.3.6 Paralelismo em Nível de Processadores

Capítulo 9

Pensamento Computacional

9.1 Bibliografia

Bibliografia Básica

- BEECHER, Karl. Computational Thinking - A beginner's guide to problem-solving and programming. Swindon, UK: BCS Learning & Development Limited, 2017. (O'Reilly) EPUB ISBN-13: 978-1-78017-36-65
- FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. Lógica de programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo: Prentice Hall, 2005. xii, 218 p. ISBN 8576050242
- MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. Algoritmos: lógica para desenvolvimento de programação de computadores. 28. ed. rev. e atual. São Paulo, SP: Érica, 2016. ISBN 9788536518657

Bibliografia Complementar

- GUEDES, Sérgio (Org). Lógica de programação algorítmica. São Paulo: Pearson, 2014. ISBN 9788543005546
- MANZANO, José Augusto N. G. Estudo dirigido de algoritmos. 15. São Paulo Erica 2011 1 recurso online ISBN 9788536519067
- SOUZA, Marcos Fernando Ferreira de. Computadores e sociedade: da filosofia às linguagens de programação. Editora Intersaberes 208 ISBN 9788559722116
- TORRES, Fernando E. et al. Pensamento computacional. Porto Alegre: SAGAH, 2019. ISBN 978-85-9502-997-2

- FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. Lógica de programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo: Prentice Hall, 2005. xii, 218 p. ISBN 8576050242

9.2 Conceitos e Competências de Pensamento Computacional

9.2.1 O que é?

9.2.2 A Importância do Pensamento Computacional

9.2.3 Visão Geral

9.2.4 Decomposição

9.2.5 Reconhecimento de Padrões

9.2.6 Abstração

9.2.7 Automação

9.2.8 Paralelização

9.2.9 Simulação

9.2.10 Avaliação de Soluções

9.3 Computação Desplugada

9.3.1 O que é?

9.3.2 A importância da Computação Desplugada

9.3.3 Compreensão de Texto

9.3.4 Adivinhação de Número

9.3.5 Roteiro Turístico

9.3.6 Nonograma

Parte II

**Análise e Projeto de
Software**

Capítulo 10

Projeto: Desenvolvimento de uma Aplicação Interativa

Capítulo 11

Algoritmos e Estrutura de Dados

Capítulo 12

Desenvolvimento Web Back-End

Capítulo 13

Design de Interação

Capítulo 14

Engenharia de Requisitos de Software

Capítulo 15

Fundamentos de Redes de Computadores

Capítulo 16

Manipulação de Dados com SQL

Capítulo 17

Modelagem de Dados

Capítulo 18

Programação Modular

Parte III

Processo de Negócio e Desenvolvimento de Software

Capítulo 19

Projeto: Desenvolvimento de uma Aplicação Móvel em um Ambiente de Negócio

Capítulo 20

Desenvolvimento de Aplicações Móveis

Capítulo 21

Estatística Descritiva

Capítulo 22

Gerência de Configuração

Capítulo 23

Gerência de Projetos de TI

Capítulo 24

Gerência de Requisitos de Software

Capítulo 25

Qualidade de Processos de Software

Parte IV

Infraestructura para Sistemas de Software

Capítulo 26

Projeto: Desenvolvimento de um Aplicação Distribuída

Capítulo 27

APIs e Web Services

Capítulo 28

Arquitetura de Software Distribuído

Capítulo 29

Banco de Datos NoSQL

Capítulo 30

Cloud Computing

Capítulo 31

Projeto de Software

Capítulo 32

Teste de Software

Parte V

Empreendedorismo e Inovação com Sistemas de Software

Capítulo 33

Projeto: Desenvolvimento de um Sistema Sociotecnológico Inovador

Capítulo 34

Compliance em TI

Capítulo 35

Computadores e Sociedade

Capítulo 36

Empreendedorismo e Inovação

Capítulo 37

Implantação de Soluções de TI

Capítulo 38

Segurança Aplicada ao Desenvolvimento de Software